

EV316937314

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

**APPLICATION FOR LETTERS PATENT**

of

**Jason P. Chalecki**

**Kelvin S. Yiu**

and

**Prakash Sikchi**

for

**Offline Editing of XML Files Using A Solution**

ATTORNEY'S DOCKET NO. MS1-1881US

## Offline Editing of XML Files Using A Solution

### RELATED APPLICATIONS.

[0001] This application is a continuation-in-part of US Patent Application Serial No. 10/395,490, filed on March 24, 2003, titled "System and Method for Offline Editing of Data Files", which is incorporated in its entirety by reference.

### TECHNICAL FIELD

[0002] This invention generally relates to editing XML files while offline.

### BACKGROUND

[0003] Extensible markup language (XML) is increasingly becoming the preferred format for transferring data. XML is a tag-based hierarchical language that is extremely rich in terms of the data that it can be used to represent. For example, XML can be used to represent data spanning the spectrum from semi-structured data (such as one would find in a word processing document) to generally structured data (such as that which is contained in a table). XML is well-suited for many types of communication including business-to-business and client-to-server communication. For more information on XML, XSLT, and XSD (schemas), the reader is referred to the following documents which are the work of, and available from, the W3C (World Wide Web consortium): XML Schema Part 2: Datatypes; XML Schema Part 1: Structures, and XSL Transformations (XSLT) Version 1.0; and XML 1.0 second edition specification.

[0004] One of the reasons that data files written in XML are often preferred for transferring data is that XML data files contain data, rather than a combination of data and the software application needed to edit the data. One problem with XML data files, however, is that to edit an XML data file, a user needs to first install a solution software application used to access, view, and edit the data file.

[0005] When a user is online, his computer can run a host application capable of accessing the Internet, such as Microsoft® Internet Explorer®, which can silently discover and deploy a solution that enables the user to author and access a corresponding XML data file.

[0006] If a user wishes to save an XML data file for later, offline use, however, the user may encounter various problems. In some cases, a user wishing to reopen an XML data file offline will not be able to do so because he can no longer discover and deploy the XML data file's solution application. The user can no longer discover a solution if he is no longer online and the solution is accessible only online. In other cases, a user can access and deploy the solution application, but to do so the user must proactively discover the solution's name and where on his computer it resides, which a user may not know. And sometimes, a user's host application discovers the solution's name and where it resides, but the user has to instruct his computer to deploy the solution application, rather than the computer automatically opening the solution file when the user attempts to open the corresponding XML data file.

[0007] Even in those cases where a user can continue to author and access an XML data file offline by actively installing the XML data file's solution application, there often is another problem. When a user instructs his host application to open a solution for an XML data file, his host application may trust the solution, thereby setting the user's computer at risk. A solution originally accessed online could, for instance, contain a virus or worm. When the user instructs his host application to install the solution, it could introduce the virus or worm.

[0008] Given the foregoing, it would be an advantage in the art to provide editing data files offline that is neither inconvenient nor dangerous.

## SUMMARY

[0009] The following description and figures describe an offline editing tool enabling offline editing of an XML data file with silent discovery and deployment of the XML data file's solution. To discover the solution, a processing instruction (PI) in an XML data file is read to determine the solution's origin. The PI contains an entity that can be a href attribute that points to a URL, a name, a target that includes the character string that identifies an application used to create an HTML electronic form associated with the XML data file, or a href attribute and at least one of a PI version and a product version. This offline editing tool enables a user to edit XML data files by performing certain actions before the user attempts to edit the XML data file while offline. When a user first opens an XML data file when online, for instance, the editing tool can download the data file's solution into a cache for later retrieval.

[0010] The offline editing tool follows appropriate security precautions to contain possibly dangerous code in an XML data file's solution even when the solution is installed from a local source, such as when a user is offline. This offline editing tool determines what level of security is appropriate for an XML data file's solution based on the original source of the solution.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The detailed description is described with reference to the accompanying figures in which the same numbers are used throughout the disclosure and figures to reference like components and features. Series 100 numbers refer to features originally found in Fig. 1, series 200 numbers refer to features originally found in Fig. 2, and series 300 numbers refer to features originally found in Fig. 3, and so on.

[0012] Fig. 1 illustrates a communications network and a system capable of implementing a method for offline editing of data files.

[0013] Fig. 2 illustrates an exemplary electronic form whereby a user can edit a data file.

[0014] Fig. 3 is a flow diagram of an exemplary process for editing of data files while online or offline.

[0015] Fig. 4 is a flow diagram of an exemplary process for offline editing of a data file with security by sandboxing the data file's solution application.

[0016] Fig. 5 is a block diagram of a computer system that is capable of supporting secure online and offline editing of data files.

#### DETAILED DESCRIPTION

[0017] The following disclosure describes an easy, simple, and secure way to access data files when offline. If a user has opened a data file first online, or if the system has otherwise received the data file's solution, a document manager application can silently discover and deploy a data file's solution. The document manager allows a user to simply select a data file to open and the document manager will open the data file with a discovered and deployed solution. The user need not discover, select, or even be aware that the data file requires a solution for the data file to be edited. After selecting the data file to open, the user can then edit and access the data file in a way very similar to how it would act and appear had the user opened the data file while online.

#### [0018] Data Files, Solutions, and Host Applications

Data files, their solutions, and a host application work together to allow a user to open and edit the data file. Data files contain little or no operable code, whereas a solution file contains presentation and logic applications. Because editing a data file requires a solution, if a user tries to open a data file without a solution, she could get an error, a prompt asking the user to open a solution, or perhaps a flat list of the data in the data file.

[0019] To view and edit a data file then, the data file's solution is needed. A data file's solution application is one or more files that, when installed, are used to enable a user to view, access, and edit the data file.

[0020] In addition to the data file and its solution, a host application is needed. This application works to enable the solution to function fully. In this description, a document manager application is described, which is capable not only of acting as a host application (allowing a solution to function properly), but can also allow a user to open a data file without actively finding and installing the data file's solution.

[0021] For discussion purposes, the system and method described herein are described in the context of a single computer, a communications network, a user-input device, and a single display screen. These devices will be described first, followed by a discussion of the techniques in which these and other devices can be used.

#### [0022] Exemplary Architecture

Fig. 1 shows an exemplary architecture 100 to facilitate online and offline editing of data files. This architecture 100 includes a computing system 102 connected to a communications network 104. The system 102 is configured to go online and communicate via the communications network 104 to gain access to non-local information sources, such as sources on an intranet or global network. Alternatively, the system 102 can remain offline, where it utilizes local resources without communicating over the communications network 104.

[0023] The computing system 102 includes a user-input device 106, a display 108 having a screen 110, and a computer 112. The user-input device 106 can include any device allowing a computer to receive a designer's preferences, such as a keyboard, a mouse, a touch screen, a voice-activated input device, a track ball, and the like. With the user-input device 106, a user can edit a data file by adding or deleting information within a

data-entry field on an electronic form, for instance. The user can use the display 108 and its screen 110 to view the data files.

[0024] The computer 112 includes a processing unit 114 to execute applications, a memory 116 containing applications and files, and a network interface 118 to facilitate communication with the communications network 104. The memory 116 includes volatile and non-volatile memory, and applications, such as an operating system 120 and a document manager application 122. The memory 116 also includes a solution 124 for a data file 126. The solution 124 is located locally in the memory 116, but often has a different original source, such as a source on the communications network 104. The solution 124 contains one or more files, such as a presentation file 128, a logic file 130, and a list file 132, which will be discussed in greater detail below.

[0025] The document manager application 122 facilitates offline editing of the data files 126 and is executed by the processing unit 114. The document manager 122 is capable of acting as a host application and enabling a user to open the data file 126 without actively finding and installing the data file's solution 124. Without any user interaction, other than the user attempting to open the data file 126, the document manager 122 discovers and installs the data file's solution 124. Thus, the user does not have to do anything but request to open the data file 126. The user does not have to discover the data file's solution 124. The user does not have to install the data file's solution 124. This silent discovery and deployment allows the user to view, edit, and otherwise interact with the data file 126 with just a single request. In addition, the document manager 122 can provide security offline similar to the security that the user typically enjoys when running a solution online.

[0026] A view of the data file 126 is depicted on screen 110 through execution of the data file's solution 124. The solution 124 contains one or more applications and/or files that

the document manager 122 uses to enable a user to edit the data file 126. To edit the data file 126 in a user-friendly way, the data file's solution 124 contains the presentation file 128, which includes an electronic form. This presentation file 128 gives the user a graphical, visual representation of data-entry fields showing previously entered data or blank data-entry fields into which the user can enter data. Data files often have one solution but each solution often governs multiple data files.

[0027] Fig. 2 shows an electronic form 200 entitled "Purchase Order", which is generated by the solution 124. This purchase order 200 contains data-entry fields in which a user can enter data. These data-entry fields map to the data file 126, so that the data entered into the form are retained in the data file 126.

[0028] This solution 124 presents an electronic form but also contains the logic file 130 that governs various aspects of the electronic form and the data file 126. In a reference number data-entry field 202, for instance, the solution 124 presents the data-entry field as a white box within a gray box, provides a description of the data desired with the text "Reference Number", and contains logic requiring that the user enter only numbers. Thus, if the user attempted to enter letters, the logic file 130 of the solution 124 would not permit the user's entry. The solution 124 could reject it and inform the user of the problem, such as with a sound, flashing error signal, pop-window, or the like.

[0029] The logic file 130 is employed in the solution 124 to ensure that the right kind of data is being entered and retained by the data file 126. A user's business manager attempting to reference purchases with a reference number, for instance, would like the solution 124 to have numbers in the reference number data-entry field 202; the manager may not be able to determine how an order should be handled if the reference number entered is incorrect because it contains letters.

[0030] Similarly, suppose a business manager wants the delivery date for delivery of a purchased product. To require this, the logic file 130 of purchase order 200's solution 124 could be constructed to require a date to be entered into a date-required data-entry field 204. The logic file 130 can be internal to the solution 124, or can be implied from the data file 126 even if the data file 126 is primarily data. The logic file 130 can also be a schema, such as an XML schema.

[0031] A solution can govern multiple data files. The exemplary purchase order 200, for example, allows one or more users to fill out many different orders. Each time a user fills out a purchase order form, the system 102 can create a separate data file for that order. Often, a user will create many different data files having the same solution. For each data file edited after the first, the system 102 is likely to have the appropriate solution stored in the memory 116. Thus, if a user previously opened a first data file and later attempts to open a second data file, both of which utilize the purchase order 200 solution, the document manager 122 can silently discover and deploy the purchase order 200 solution to enable the user to edit the second data file. How the document manager 122 discovers and deploys solutions will be discussed in greater detail below.

[0032] A solution can be one file or contain many files, so long as the files used to edit data files it governs are included. The solution 124 of Fig. 1 includes the listing file 132, which is a manifest of all of the other files in the solution 124 and contains information helping the document manager 122 to locate them. The logic file 130 and presentation file 128 can be joined or separate. The presentation file 128 helps the document manager 122 present or give a view of a form enabling entry of data into the data file 126, such as a visual representation of the data file 126 by the purchase order 200 electronic form. In some implementations, the presentation file is an XSLT file, which, when applied to an XML data file, generates a XHTML (eXtensible Hyper-Text Markup Language) or

HTML (Hyper-Text Markup Language) file. XHTML and HTML files can be used to show a view on the screen 110, such as the purchase order 200 of Fig. 2.

[0033] A solution, such as the solution 124, can also include various files or compilations of files, including a manifest file setting forth names and locations for files that are part of the solution 124. The files within the solution 124 can be packaged together, or can be separate. When separate, the list file 132 acts as a manifest of the files within the solution 124. The list file 132 can also include other information, such as definitions, design time information, data source references, and the like. When the files are packaged together, the document manager 122 can simply install and execute the packaged solution file for a particular data file. When not packaged, the document manager 122 can read the list file 132, find the listed files, and install and execute each of the listed files for the particular data file. The list file 132 and the packaged solution file can be interrelated in that a packaged file contains the list file 132 and the list file 132 lists files packaged within the packaged file, although usually only one need be discovered by the system 102 to open a particular data file.

[0034] Like solutions, data files can come in various types and styles. As mentioned above, data files can be written in XML or some other mark-up language, or can be written in other languages. Most data files, however, do not contain extensive logic and other files or code. One of the benefits of having data files separate from their solutions, is that it makes the data within them easier to mine. Because the data files are separate from their solution, the document manager 122 makes them easy to open and edit by silently discovering and deploying the solution for the data file.

[0035] Data files also are typically concise and data-centered so that the data they contain can be more easily accessed or manipulated by multiple software applications, including software not typically used in a solution, such as an application that searches for a

particular type of data and compiles that data into a report. A non-typical application, for example, could be one that compiles a report of all of the purchase orders required to be mailed by a certain date by searching through and compiling the data entered into data files through the date required data-entry field 204 of the purchase order 200 electronic form.

[0036] The above devices and applications are merely representative, and other known devices and applications may be substituted for or added to those shown in Fig. 1. One example of another known device that can be substituted for those shown in Fig. 1 is the device shown in Fig. 5.

#### [0037] Techniques for Silent Discovery and Deployment of Data File Solutions

##### Overview

Fig. 3 shows a process 300 for silently discovering and deploying a data file's solution. The process 300 is illustrated as a series of blocks representing individual operations or acts performed by the architecture 100. The process 300 may be implemented in any suitable hardware, software, firmware, or combination thereof. In the case of software and firmware, the process 300 represents a set of operations implemented as computer-executable instructions stored in memory and executable by one or more processors.

#### [0038] Silent Discovery and Deployment

At block 302, the system 102 receives input from a user to open the data file 126. The user may simply click on an icon representing the data file 126 or otherwise select the data file 126 after which the system 102 opens the data file 126.

[0039] At block 304, the system 102 discovers a solution identifier in the selected data file 126. This assumes that the data file 126 is one in which the document manager 122 is capable of reading. The document manager 122 can read data files created at some

previous time by the user's or another's document manager 122. In one implementation, the document manager 122 can also read the data file 126 if it is created by another application that builds a solution identifier into the data file 126.

[0040] This solution identifier can give the system 102 an original source for the solution 124. With an original source for the solution 124, the system 102 has one manner in which to help determine the proper security appropriate for the solution 124. How the system 102 and the document manager 122 handle security for a solution 124 is set forth in greater detail below.

[0041] The solution identifier is typically a URL (Uniform Resource Locator) or URN (Uniform Resource Name), but can include other types of names and/or locators. URLs give locations and URNs give names of resources, such as the solution 124, which are typically accessible through the communications network 104. With the solution identifier, the system 102 can determine the original source for the solution 124 (where it first came from) and whether or not the system 102 has seen the solution 124 before.

[0042] In one implementation, the solution identifier is part of a processing instruction included within the data file 126. This processing instruction is often part of data files and can include various instructions to host applications, such as the document manager 122. Processing instructions, while not strictly data, do not rise to the level of an applet or application typically included in a solution for a data file. For data files written in XML, for instance, the processing instructions are usually not written in XML, but rather are just a piece of information commonly included. A processing instruction in an XML data file can look like

```
<? mso-infoPathSolution
solutionVersion="1.0.0.3"
PIVersion="1.0.0.0"
href="http://xdsp04-neten/MiladinP/Forms/template.xsn" ?>
```

where:

solutionVersion is the version number of the solution used to generate the XML data (e.g., XML document);  
PIVersion is the version of the processing instruction;  
href is a Universal Resource Name (URN) to the solution; and  
the URN represents the solution.

This processing instruction gives the document manager 122 a solution identifier, which here gives the original source for the solution for the data file. This solution identifier includes the URN indicating that the original location for the solution is at a remote server accessible by accessing the communications network 104 through the network interface 118.

**[0043]** In another implementation, the document manager application 122 determines a solution that is to be used with XML data, such as the data file 126. The solution is determined by locating a Processing Instruction (PI) within the XML data. When the PI is located in the XML data, the payload of the PI is examined to determine the solution that is to be used with the XML data. A Universal Resource Locator (URL) in the PI can be used to identify a location of the solution. The solution can then be retrieved from its location. Once retrieved, the solution can be used to present a visual appearance of the XML data, to receive interactive edits to the XML data from an editing user, and then to update both the visual appearance and the XML data using the edits to the XML data from the editing user. An example of a PI in the XML data that uses a URL for discovery of its corresponding solution is:

```
<?mso-InfoPathSolution  
href="URL to solution"  
name="solution URN"  
version="version number"?>
```

where:

href is a URL to the solution;  
name is a Universal Resource Name (URN) representing the solution; and

version is the version number of the solution used to generate the XML data (e.g., XML document).

[0044] Another example of a PI that identifies its solution by location is:

```
<?mso-infoPathSolution
href="file:///\\ndpublic\public\user-alias\Solution\manifest.xsf"
PIVersion="1.0.0.0"
solutionVersion="1.0.0.90"
productVersion="11.0.4920" ?>
```

In this example, the PI has five (5) parts. The first part is the name of the PI which is 'mso-infoPathSolution'. The second part is the 'href' that is a URL that specifies the location of the solution. Stated otherwise, the 'href' is a pseudo-attribute of the PI that contains a URL that specifies the location of the solution. The URL, in different alternatives, could refer to an 'http', to a file, or to an 'ftp', in accordance with known file path conventions. Additionally, relative URLs may also be used. The third part is the version of the PI ('PIVersion'). The PIVersion pseudo-attribute is used to identify the version of the PI itself. The fourth part is the 'solutionVersion' which is a pseudo-attribute used to identify the version number of the solution used to generate the corresponding XML document. The fifth part is the productVersion pseudo-attribute that is used to identify the version of the document manager application 122 that created the XML document for which the solution is to be used. In this example, the document manager application 122 is the InfoPath™ software application provided by Microsoft Corporation of Redmond, WA, USA. Also in this example, the third through fifth parts have the format of 'x.x.x.x'.

[0045] The foregoing implementations provide two examples in which a URL in the PI could be used to identify a location of the solution to be used with an XML document. In yet another implementation, a PI in XML data contains a name from which the solution for the XML data can be discovered. Stated otherwise, the PI in the XML data identifies its solution by name. In this case, the document manager application 122 can determine

the solution to use with XML data (e.g., the payload) by the schema of the top level node of the structured nodes in the corresponding XML data. As in prior implementations, the document manager application 122 performs a process in which the PI is located and examined. An example a PI that identifies a solution by name is:

```
<?mso-infoPathSolution
name="urn:schemas-microsoft-com:office:infopath:oob:AbsenceRequest:1033"
PIVersion="1.0.0.0"
solutionVersion="1.0.0.0"
productVersion="11.0.4920" ?>
```

In this example, the PI has five (5) parts. The first part is the name of the PI which is 'mso-infoPathSolution'. The second part is a name pseudo-attribute that refers to a URN which is the name of the solution. Such a solution is typically registered with the document manager application 122 before the document is opened, and the document manager application 122 looks it up by name in its list of available solutions. The third through fifth parts are identical to the immediately preceding example. As in the immediately preceding example, the present example also represents that the document manager application 122 is the InfoPath<sup>™</sup> software application provided by Microsoft Corporation of Redmond, WA, USA.

[0046] In a still further implementation, the document manager application 122 can determine an online sandboxed solution that is to be used with an XML document. To do so, a PI in the XML document is located. An example of a PI for an online sandboxed solution is:

```
<?mso-xDocsSolution
version="1.0.0.20"
href="http://po.car-company.com/po/po.xsf"?>
```

In this example, the href contains a solution identifier for which the solution has a path with the suffix '.xsf'.

[0047] In yet another implementation, the document manager application 122 can determine a single file online sandboxed solution that is to be used with an XML document. To do so, a PI in the XML document is located. An example of a PI for a single file online sandboxed solution is:

```
<?mso-xDocsSolution version="1.0.0.20"
href="http://po.car-company.com/po/po.xsn"?>
```

In this example, the href contains a solution identifier for which the solution has a path with the suffix '.xsn'.

[0048] In another implementation, the document manager application 122 can determine the solution that is to be used with a preexisting electronic form template. To do so, a PI in the XML document is located. An example of a PI for such a case is:

```
<?mso-xDocsSolution
name="urn:schemas-microsoft-
com:office:InfoPath:oob:C9956DCEE66F8924AA31D467D4F366DC"
version="1.0.0.20"?>
```

In this example, the URN in the 'name' attribute is preceded by "InfoPath:oob" to designate that the template is provided with the InfoPath™ software application provided by Microsoft Corporation of Redmond, WA, USA.

[0049] In a still further implementation, the document manager application 122 can determine a trusted solution that is to be used with XML data. To do so, a PI in the XML data is located. An example of a PI for a trusted solution is:

```
<?mso-xDocsSolution
name="urn:schemas-microsoft-com:office: po. car-company.com"
version="1.0.0.20"?>
```

In this case, the document manager application 122 can determine the solution to use with XML data (e.g., the payload) by the schema of the top level node of the structured nodes in the corresponding XML data. The solution is 'trusted' in the sense that a logical

relationship is established between domains to allow pass-through authentication, in which a trusting domain honors the logon authentications of a trusted domain.

[0050] One of the advantages of the document manager 122 is that it enables a user to open the data file 126 without the user needing to discover the data file's solution 124, install the solution 124, or even know that the solution 124 exists. This enables users to open data files simply and easily, and in many cases enables them to edit a data file offline that they would otherwise not have been able to edit.

[0051] With the solution identifier, the system 102 computes a special name for the solution 124 (block 306). This special name is designed to be a name easily found only by the document manager 122. The special name, because it is computed and findable by the document manager 122 but is not intended to be discoverable by other applications, allows for greater security in downloading possibly hostile solutions from the communications network 104.

[0052] In one implementation, the document manager 122 takes the solution identifier and computes a unique special name for the solution identifier. This unique special name is repeatable; the next time the document manager 122 computes a unique special name for the same solution identifier, the same unique special name will be created. By so doing, the document manager 122 can find a previously downloaded solution by computing the unique, special name and then searching for the unique, special name to determine if the solution is available locally for offline use (such as by having the solution stored in the memory 116).

[0053] In another implementation, the document manager 122 computes a unique special name by computing a hash, such as a Message Digest 5 hash (MD5 hash), of the solution identifier. By computing a one-way hash of the solution identifier, the document manager 122 creates a unique, special name that is a file of 128 bits from the digits of the

solution identifier. The MD5 hash can be computed by knowing the URL. In one implementation, a cache for a solution can be structured as:

%AppData%\InfoPath\Solution Cache\[16 character random GUID]\[MD5 hash  
of URL or URN

, where the random GUID protects the cache from other applications.

**[0054]** The system 102 uses the special name, which corresponds to a solution identifier and thus the data file's solution 124, to search through locally accessible sources for the solution 124 (block 308). The system 102 may, for instance, search files and folders in the memory 116 of Fig. 1 for files and/or folders with the same name as the special name computed in the block 306.

**[0055] When the Special Name is Found**

If the system 102 finds the special name (i.e., the "Yes" branch from block 310) the solution 124 was saved earlier in the system 102 searched locally in the block 308. Thus, when the special name is found, the system 102 knows that the solution 124 referred to in the data file (which the user is attempting to open) is accessible offline by the system 102. The solution 124 is usually stored in the memory 116 but can be stored in other searchable, local sources that the system 102 does not have to go online to find.

**[0056]** The solution 124, stored at the source and found using the special name, may not be current, however. Because of this, the system 102 determines whether or not the system 102 is online or offline (block 312). If online (i.e., the "Yes" branch from block 312), the system 102 will attempt to determine whether or not a more up-to-date solution should be installed (discussed below); if offline, the system 102 will proceed to install the locally stored solution 124 (block 314).

**[0057] If the Solution is Found and the System is Offline**

If the solution 124 is found and the system 102 is offline, the system 102 proceeds to install the solution 124 from the memory 116 or another locally accessible source (block 314).

**[0058]** The system 102 installs the solution 124 silently in that the user does not need to know that the solution 124 was discovered, found, or was being installed. Thus, the system 102 enables a user to edit the data file 126 when offline by silently discovering and deploying the data file's solution 124.

**[0059]** In one implementation, the system 102 installs the solution 124 and then opens the data file 126 in such a manner as to mimic how the data file 126 would be opened had the user opened the data file 126 with the solution accessible online, such as through opening the data file 126 with Microsoft® Internet Explorer®. The system 102 does so to make opening and editing the data file 126 as comfortable for the user as possible, because many users are familiar with opening data files online. One possible difference, however, is that if the system 102 has a slow connection to the communications network 104, the document manager 122, by installing the solution 124 from a local source like the memory 116, may more quickly open the data file 126 than if the user were online.

**[0060]** Also in block 314, the document manager 122 can install the solution 124 for the selected data file with certain constraints for security, which will be discussed in greater detail as part of a process 400 of Fig. 4.

**[0061]** In block 316, the system 102 opens the data file 126 to enable the user to edit the data file 126. One example of an opened data file (and solution) enabling edits is the purchase order 200 of Fig. 2. In this example, the user is able to edit the data file 126 by adding, deleting, or changing data in data entry fields (like the reference number data-entry field 202 and the date required data-entry field 204) even though offline.

[0062] Following the previous blocks, a user can easily open a data file offline without having to discover or deploy the data file's solution. This enables users, for example, after first opening a solution online, to open a data file offline. A user can open a data file online and edit it by adding a reference number through the reference number data-entry field 202 of the purchase order 200 electronic form and then stop editing the data file (the data file would contain the added reference number by the system 102 adding the reference number to the data file). The user could then go offline, such as by taking his or her laptop on a business trip, and complete filling out the electronic form. Or the user could send the partially filled-out data file to another user to fill out the rest of the electronic form, which the other user could do so long as the other user's system contains a stored solution. This flexibility allows users and businesses a greater ability to use information by keeping data and solutions separate and by allowing offline use of data files.

**[0063] If the Solution is Found and the System is Online**

Assuming the system 102 finds the special name and the system is online, the system 102 will attempt to determine whether the current solution is the most recent version or a more up-to-date solution is available. In block 318, the system 102 compares the time stamp of the stored solution 124 and the online solution. Since the system 102 is online, it can access the solution (here we assume that the original origin of the solution 124 is from an online source). If the solution identifier from the data file 126 selected by the user contains a reference to the solution 124 being accessible online, the system 102 goes online to check whether or not the online solution is newer than the stored solution 124 (block 320). In one implementation, the system 102 compares the time stamp of the online solution with a time stamp on the stored solution 124.

[0064] If the online solution is not newer (i.e., the “No” branch from block 320), the system 102 proceeds to the block 314, installing the stored solution 124. If the online solution is newer than the stored solution 124 (i.e., the “Yes” branch from block 320), the system 102 either replaces the stored solution 124 with the online solution or otherwise updates the older, stored solution 124.

**[0065] Downloading the Solution for Later Use**

In block 322, the architecture 100 (or the system 102 by accessing the communications network 104) downloads a solution into a locally accessible source such as the memory 116. The system 102 downloads this solution when the data file 126 selected by a user contains a solution identifier for a solution for which the system 102 does not have local access (such as it not being cached) or for which the system 102 has local access but the cached or stored version of the solution (the solution 124) is older than the online version.

[0066] In either case, the system 102 has already discovered the solution identifier for the solution and computed a special name for the solution. The system 102 then downloads the solution from the online source and saves it into a folder named with the special name (block 324). If a solution already exists in that folder, the system 102 replaces it with the newer version or otherwise updates the currently cached solution. The resulting new or updated version then being the solution 124.

[0067] In one implementation, the system 102 saves the solution to a unique location within the system 102’s accessible memory. The system 102 does so in cases where the system 102 is used by multiple users. By so doing, the system 102 is able to determine which of the users that use the system 102 or load files into memory locally accessible by the system 102 saved the particular solution. Also by so doing, the system 102 may provide greater security for the computer 112 and its users.

[0068] Process 300 in Fig. 3 allows for editing of data files while online or offline. In each of the various implementations discussed above in the context of Fig. 3, the document manager application 122 performed a process in which a PI is located and examined. Stated otherwise, the document manager application 122 performs a process that searches XML data to find a solution of the XML data. In other implementations, the search by document manager application 122 may examine various storage locations for the solution, such as file storage on a network resource that is in communication with a computing device that is executing the document manager application 122. Alternatively, the search may examine local cache in a computing device that is executing the document manager application 122. If the search by the document manager application 122 in the XML data does not locate a PI that specifically identifies the document manager application 122, then a diagnostic can be output, such as by way of a visibly displayed descriptive error. If a PI is found in the search that specifically identifies the document manager application 122, then it is determined if the PI contains a 'href' attribute. If the 'href' attribute is in the PI, then a computation is made of the name of the folder in the solution cache using the URL associated with the 'href' attribute. A look up is first performed on the list of locally cached solutions using the URL in the 'href' attribute (e.g., where the solution is 'online sandboxed'). If, however, the solution is not located in the list of locally cached solutions, then the solution must be retrieved from another resource. In this case, the solution can be downloaded from the URL associated with the 'href' attribute in PI.

[0069] In another implementation, where the solution is located in the list of locally cached solutions, it can be ascertained whether the most recent version of the solution has been obtained. To do, a version, electronic tag, or time stamp in the PI can be examined, such as by an auto-update process. If the examination indicates a change in the original

solution (e.g., a server copy of the solution is different), then the latest solution can be downloaded. For instance, the document manager application 122 can be configured to explode the solution to a temporary location within the locally cached solutions and then can compare the version of the downloaded solution to the version of the current cached copy of the solution.

[0070] If the search by the document manager application 122 finds a PI but fails to locate the solution in the list of locally cached solutions or by download, then a diagnostic can be output, such as by way of a visibly displayed descriptive error. If the search by the document manager application 122 finds a PI that contains a name attribute associated with a URN, a computation can be performed of the name of the folder in the solution using the URN. A lookup can then be performed on an HTML installed solution using the name attribute in a catalog of solutions. If the solution is thereby found, a comparison can be made between the cached version of the solution and the original version of the solution. This comparison is performed to make sure that the cached version of the solution is up-to-date. In the case of the InfoPath™ software application provided by Microsoft Corporation of Redmond, WA, USA, through the Microsoft Office™ software application, if the name is prefixed by a character string that includes “InfoPath” and “oob”, then the XML data located by the application is referring to an InfoPath template or out-of-the-box (OOB) solution. In such a case, an assumption might be made that solutions for the InfoPath templates are installed at a location shared by all users of a specific computing device (e.g., C:\Program Files\Microsoft Office\Templates).

[0071] The steps set forth in process 300 in Fig. 3 for discovering and using a solution are described above for various implementations. Still further implementations can be used by the document manager application 122 to discover and use a solution. In these

further implementations, the document manager application 122 attempts to associate an XML document with a solution by using a href attribute in a PI in the XML document that points to a URL. In yet other implementations, the document manager application 122 initiates a search in an XML document for a PI having a href attribute, a name, and a version. When the document manager application 122 finds such a PI, the XML document can be associated with a solution through the use of the href attribute (e.g., a URL).

**[0072]** When the InfoPath™ software application provided by Microsoft Corporation of Redmond, WA, USA, through the Microsoft Office™ software application is used to design an electronic form, a PI in an XML document corresponding to the electronic form will have various characteristics. As such, any document manager application 122 that is attempting to discover a solution for an electronic form that was created by the InfoPath™ software application should be advantageously configured to locate the solution by searching for a PI in the corresponding XML document that has one or more of these various characteristics. When so discovered, the document manager application 122, which may be different than the InfoPath™ software application, can then be used to view and/or interactively enter/change XML data through the electronic form that was created by the InfoPath™ software application.

**[0073]** Once such characteristic of an XML document corresponding to an electronic form that was designed, created, or changed by the InfoPath™ software application is that it has a PI containing the solution identifier that will most likely be the first PI in the XML document corresponding to the electronic form. Another such characteristic is that the PI containing the solution identifier contains both a href attribute and a character string that identifying the PI version and/or the product version of the InfoPath™ software application. The href attribute need not be the first pseudo-attribute in the PI, but could

be anywhere in the PI. As such, the PI version could be first in the PI followed by the href attribute. Yet another characteristic is that the target of the PI containing the solution identifier also contains the character string “mso-InfoPathSolution” solution, where that character string is the first character string in the PI and could be followed by a href attribute. Yet another characteristic is that the PI containing the solution identifier contains a URL or URN that will most likely point to a path having a suffix that is either ‘\*.xsf’ (e.g., a manifest or listing other files) or ‘\*.xsn’ (a file that contains multiple files compressed into one file and that are extractable with the extract ‘\*.exe’ utility).

[0074] Given the foregoing characteristics of electronic forms designed, created, or changed using the InfoPath™ software application, the document manager application 122 can be configured to discover the solution of an electronic form by simply finding the first PI in the XML document. Alternatively, the document manager application 122 can be configured to discover the solution by examining a PI in the XML document for the electronic form to see if it includes a character string that is likely to represent a solution corresponding to the XML document. When the likelihood is high, the name can be used to discover the solution. As a further alternative, the document manager application 122 can be configured to discover the solution by examining the target of a PI in the corresponding XML document for the character string “mso-InfoPathSolution”, and/or by looking for a URL in the PI having a suffix of \*.xsf or \*.xsn. As yet another alternative, the document manager application 122 can be configured to discover the solution by trying, as a potential solution, each name that is in quotation marks in the PI and is associated with a href attribute in a PI in the XML document, and then using the solution that is returned after a success following one or more of such attempts. As mentioned above, the href attribute need not be the first pseudo-attribute in the PI, and the document manager application 122 may take this into account when assessing the PI

for its likelihood of containing the correct solution identifier. In a still further another alternative, the document manager application 122 can be configured to discover the solution by trying as a potential solution each URL in each PI of an XML document, and then using the solution that is returned after a success following one or more of such attempts. As noted above, the URL or URN in the successful PI will most likely point to a '\*.xsf' or '\*.xsn' path, and the document manager application 122 may take this into account when assessing the PI for its likelihood of containing the correct solution identifier.

**[0075] Techniques for Secure Deployment of Data File Solutions**

Fig. 4 shows a process 400 for making deployment of data file solutions more secure. The process 400 is illustrated as a series of blocks representing individual operations or acts performed by the system 102. The process 400 may be implemented in any suitable hardware, software, firmware, or combination thereof. In the case of software and firmware, the process 400 represents a set of operations implemented as computer-executable instructions stored in memory and executable by one or more processors.

**[0076]** Through the process 300, discussed above, the system 102 enables a user to open and edit a data file by silently discovering and deploying the data file's solution. In the process 400, the system 102 acts to protect the user from the solution because some solutions contain dangerous code, like viruses and worms. To help prevent dangerous code from damaging the user's files and/or computer, the system 102 sandboxes the solution, if appropriate.

**[0077]** As part of this security, the system 102 can be configured to ask a user during an attempt to open a data file how the user wants to sandbox the data file's solution. Many

users, however, often do not know what level of sandboxing to chose, and thus the explicit prompting slows the process and consumes unnecessary user time.

[0078] To make opening and editing a data file as easy as possible and because many users do not know what level of sandboxing a solution should be run within, the system 102 can be configured to sandbox a solution automatically, as set forth in Fig. 4 and the process 400 below.

[0079] In block 402, the system 102 determines the origin of a solution, such as the solution 124 of Fig. 1. The system 102, from block 304, discovered the solution identifier from the data file 126. With this solution identifier, the system 102 can determine the origin of the solution 124.

[0080] As discussed above, the solution identifier can be a URL, a URN, or another Uniform Resource Identifier (URI). Either of these can be used to locate and find the source of a solution. URLs usually indicate a non-local, online source for a solution like a remote server accessible through the communications network 104. URNs give the name of a solution, which typically can be accessed online (but could be accessed from a local, offline source), and are less subject to change by those in control of the solution. In either case, these solution identifiers give the system 102 the original source of the solution. The solution may be cached or otherwise stored by the system 102 in a local source (such as the solution 124 in the memory 116), but the local source is not indicated as the solution's origin by the solution's solution identifier, the local source is a new source.

[0081] With the origin of the solution known, the document manager 122 sets the appropriate level of security for the solution 124, sandboxing the solution 124 based on its origin (block 404). The document manager 122 sandboxes the solution 124, if from unknown or untrustworthy sources on the global internet with a sandbox allowing the

solution 124 very little leeway in the operations it can perform, such as reading or altering other files on the computer 112. Greater trust, and so a weaker sandbox, are used for the solution 124 if it has an origin from known and more trustworthy sources accessed remotely, such as from MSN.com or a company intranet source. Very high trust, requiring a weak or no sandbox, is used for the solution 124 if it has an origin on the computer 112, such as when the user or another person using the user's computer created the solution 124.

**[0082]** Setting security levels and appropriate use of sandboxes for online use of solutions for data files are performed by various internet-capable host applications. The document manager 122, however, opens solutions and sandboxes them based on their origin even when the solution is opened from a local source that differs from the origin of the solution. Typically, internet-capable host applications, such as Internet Explorer®, will not open a solution from a local source silently, but will return an error if the online copy is not available. The document manager 122, however, opens and executes, without user interaction, solutions in appropriate sandboxes even when a solution is loaded from a local source.

**[0083]** In block 406, the system 102 installs the solution 124 within an appropriate sandbox. This appropriate sandbox limits the operations that the solution 124 can perform, thereby helping to protect the computer 112 and its files from the solution 124.

**[0084]** A Computer System

Fig. 5 shows an exemplary computer system that can be used to implement the processes described herein. Computer 542 includes one or more processors or processing units 544, a system memory 546, and a bus 548 that couples various system components including the system memory 546 to processors 544. The bus 548 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a

peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 546 includes read only memory (ROM) 550 and random access memory (RAM) 552. A basic input/output system (BIOS) 554, containing the basic routines that help to transfer information between elements within computer 542, such as during start-up, is stored in ROM 550.

[0085] Computer 542 further includes a hard disk drive 556 for reading from and writing to a hard disk (not shown), a magnetic disk drive 558 for reading from and writing to a removable magnetic disk 560, and an optical disk drive 562 for reading from or writing to a removable optical disk 564 such as a CD ROM or other optical media. The hard disk drive 556, magnetic disk drive 558, and optical disk drive 562 are connected to the bus 548 by an SCSI interface 566 or some other appropriate interface. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for computer 542. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 560 and a removable optical disk 564, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

[0086] A number of program modules may be stored on the hard disk 556, magnetic disk 560, optical disk 564, ROM 550, or RAM 552, including an operating system 570, one or more application programs 572 (such as the document manager application 122), other program modules 574, and program data 576. A user may enter commands and information into computer 542 through input devices such as a keyboard 578 and a pointing device 580. Other input devices (not shown) may include a microphone,

joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 544 through an interface 582 that is coupled to the bus 548. A monitor 584 or other type of display device is also connected to the bus 548 via an interface, such as a video adapter 586. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

[0087] Computer 542 commonly operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 588. The remote computer 588 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 542. The logical connections depicted in Fig. 5 include a local area network (LAN) 590 and a wide area network (WAN) 592. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

[0088] When used in a LAN networking environment, computer 542 is connected to the local network through a network interface or adapter 594. When used in a WAN networking environment, computer 542 typically includes a modem 596 or other means for establishing communications over the wide area network 592, such as the Internet. The modem 596, which may be internal or external, is connected to the bus 548 via a serial port interface 568. In a networked environment, program modules depicted relative to the personal computer 542, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0089] Generally, the data processors of computer 542 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the blocks described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described herein.

[0090] For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

#### [0091] Conclusion

The above-described system and method enables a user to edit data files when offline by discovering and deploying the data file's solution application. Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed invention.